REMARKS

Reconsideration of the present application is requested. Claims 1, 18, 21 and 24 have been amended.

On April 20, 2007, the Board of Patent Appeals and Interferences (hereinafter "the Board") affirmed the rejection of claims 1-5, 14 and 16-24 rendered obvious under 35 U.S.C. §103(a) by Bruckman (U.S. Patent Publication No. 2002/0051466), in view of Applicants' Admitted Prior Art (AAPA), and Tiedmann, Jr., et al. (U.S. Patent No. 5,914,950) and the rejection of claims 6-13 rendered obvious under 35 U.S.C. § 103(a) by Bruckman, AAPA, Tiedmann, and further in view of Buchholz (U.S. Patent No. 5,337,313).

In affirming the rejection, the Board dismissed Applicants' arguments that "puncturing," includes dropping, removing or deleting a portion of bits in the channel coded encoder packet to create a channel encoded encoder sub-packet because such a recitation was not explicit in the claims.[1] In response, Applicants have amended claim 1 to recite "wherein puncturing including the moving bits from the channel coded encoder packet." Although no explicit support for such an amendment is set forth in Applicants' Specification, such an amendment is inherently supported by the well-known term "puncturing."

As evidenced by the attached article entitled "Punctured Convolutional Codes of Rate (N-1)/N and Simplified Maximum Likelihood Decoding," by J. Cain et al. from 1979 (hereinafter "the Cain Article"), puncturing, as interpreted

---

[1] *Ex Parte Das*, Appeal No. 2007-0843, p. 7 (BPAI Apr. 20, 2007).

by one of ordinary skill, includes removing or deleting bits from a code.[2] Because such a term is so well-known, the amendments made herein are fully supported by "puncturing," as used in Applicants' Specification.

As noted above, claims 1-5, 14 and 16-24 stand rejected under 35 U.S.C. § 103(a) in view of Bruckman (U.S. Patent Publication No. 2002/0051466, hereinafter Bruckman), in view of Applicants' Admitted Prior Art (hereinafter AAPA), and Tiedmann, Jr., et al. (U.S. Patent No. 5,914,950, hereinafter Tiedmann).

The Examiner relies upon Bruckman to teach the "puncturing and/or repeating," step set forth in claim 1. Applicants disagree.

As shown in FIG. 1 of Bruckman, a transmitter includes packet sources 26, which may generate streams of data packets for transmission over channel 25. The dynamic packet fragmenter 28 determines fragment sizes into which packets are to be divided.[3] When an input packet from the source 26 exceeds the determined fragment size, fragmenter 28 divides the packet for transmission into multiple fragments.[4]

Applicants respectfully submit Bruckman fails to teach or suggest at least, "puncturing and/or repeating channel coded packets," wherein the "puncturing," includes "removing bits from the channel coded encoder packet," as now set forth in claim 1. Instead, at most, Bruckman arguably discloses the

---

[2] *See, e.g., Cain,* p. 97, right column, last paragraph ("In this example an R=2/3 code was constructed by periodically deleting bits from an R=1/2 code, or in other words by puncturing the code.").
[3] *Bruckman,* para. [0026], ll. 6-8.
[4] *Id.* at ll. 8-11.

fragmenting (dividing) of packets into pieces based on a transmission rate over a channel 25. As the Examiner will appreciate, segmenting or fragmenting does not include deleting or removing bits from a code. Moreover, as previously argued, fragmenting is not "repeating," as set forth in claim 1.

The Examiner correctly recognizes that Bruckman fails to teach all features set forth in claim 1, and relies upon AAPA and Tiedmann to make up for the recognized deficiencies. Neither AAPA nor Tiedmann, however, teach or fairly suggest at least, "puncturing and/or repeating channel coded packets," wherein the "puncturing," includes "removing bits from the channel coded encoder packet," as now set forth in claim 1. Therefore, even assuming *arguendo* the Examiner's combination could be made (which Applicants still do not admit), the combination of references fails to teach or fairly suggest all features of claim 1.

For at least the foregoing reasons, claim 1 is patentable over Bruckman, AAPA and/or Tiedmann. Claims 18, 21 and 24 are also patentable over the Examiner's combination of references for at least reasons somewhat similar to those set forth above with regard to claim 1. Claims 2-5, 14 and 18-20 and 22-23 are patentable at least by virtue of their dependency from claims 1, 18 or 21.

Claims 6-13 stand rejected under 35 U.S.C. § 103(a) as allegedly unpatentable over Bruckman, AAPA, Tiedmann and further in view of Buchholz (U.S. Patent No. 5,337,313). This rejection is respectfully traversed in that even assuming *arguendo* Bruckman, AAPA and/or Tiedmann could be

combined with Buchholz (which Applicants do not admit for at least the reasons somewhat similar to those set forth above), Buchholz still fails to make up for at least the deficiencies of Bruckman, AAPA, and Tiedmann with respect to claim 1. Therefore, claims 6-13 are patentable over Bruckman, AAPA, Tiedmann and/or Buchholz.

CONCLUSION

Accordingly, in view of the above amendments and remarks, reconsideration of the objections and rejections and allowance of each of claims 1-14, and 16-24 in connection with the present application is earnestly solicited.
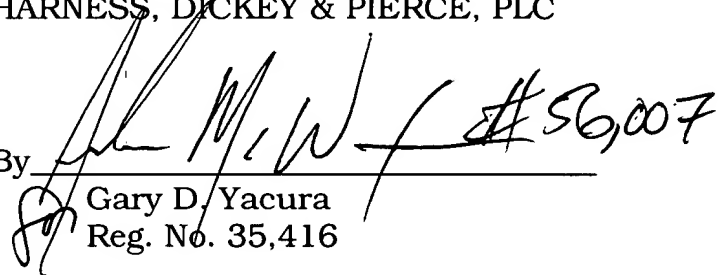
If the Examiner believes that personal communication will expedite prosecution of this application, the Examiner is invited to telephone Andrew M. Waxman, Reg. No. 56,007, at the number of the undersigned listed below.

If necessary, the Commissioner is hereby authorized in this, concurrent, and future replies to charge payment or credit any overpayment to Deposit Account No. 08-0750 for any additional fees required under 37 C.F.R. §§ 1.16 or 1.17; particularly, extension of time fees.

Respectfully submitted,

HARNESS, DICKEY & PIERCE, PLC

By _____ #56,007

Gary D. Yacura
Reg. No. 35,416

GDY/AMW:krm

P.O. Box 8910
Reston, VA 20195
(703) 668-8000

Attachments: Cain Article

[3] R. Gold, "Optimal binary sequences for spread spectrum multiplexing," *IEEE Trans. Inform. Theory*, vol. IT-13, pp. 619–621, Oct. 1967.

[4] ——, "Maximal recursive sequences with 3-valued recursive cross-correlation functions," *IEEE Trans. Inform. Theory*, vol. IT-14, pp. 154–156, Jan. 1968.

[5] S. W. Golomb, *Shift-Register Sequences*. San Francisco: Holden-Day, 1967.

[6] D. Graupe, *Identification of Systems*. Huntington, New York: R. E. Krieger Publishing Co., 1976.

[7] T. Helleseth, "Some results about the cross-correlation function between two maximal linear sequences," *Discrete Mathematics*, vol. 16, pp. 209–232, Nov. 1976.

[8] D. A. Huffman, "The generation of impulse-equivalent pulse trains," *IRE Trans. Inform. Theory*, vol. IT-8, pp. S10–S16, September 1962.

[9] T. Kasami, "Weight distribution formula for some class of cyclic codes," Coordinated Science Laboratory, University of Illinois, Report R-285, Apr. 1966.

[10] ——, "Weight distribution of Bose–Chaudhuri–Hocquenghem codes," in *Combinatorial Mathematics and Its Applications*, R. C. Bose and T. A. Dowling (eds.) Chapel Hill, NC: Univ. of North Carolina Press, 1969.

[11] F. J. MacWilliams, "An example of two cyclically orthogonal sequences with maximum period," *IEEE Trans. Inform. Theory*, vol. IT-13, pp. 338–339, Apr. 1967.

[12] F. J. MacWilliams and N. J. A. Sloane, "Pseudo-random sequences and arrays," *Proceedings of the IEEE*, vol. 64, pp. 1715–1729, Dec. 1976.

[13] K. A. Meshkovskii, "A new class of pseudorandom sequences of binary signals," *Probl. Peredach. Inform.*, vol. 9, pp. 117–119, Jul.–Sept. 1973.

[14] M. B. Pursley and D. V. Sarwate, "Performance evaluation for phase-coded spread-spectrum multiple-access communication—Part II: Code sequence analysis," *IEEE Trans. Comm.*, vol. COM-25, pp. 800–803, Aug. 1977.

[15] N. Zierler, "Linear recurring sequences," *J. Soc. Ind. Appl. Math.*, vol. 7, pp. 31–48, 1959.

Punctured Convolutional Codes of Rate $(n-1)/n$ and Simplified Maximum Likelihood Decoding

J. BIBB CAIN, MEMBER, IEEE, GEORGE C. CLARK, JR., MEMBER, IEEE, AND JOHN M. GEIST, MEMBER, IEEE

Abstract—The structure of punctured convolutional codes is described, and it is indicated how their use simplifies the design of maximum likelihood decoders. The best codes of this class for rates 2/3 and 3/4 are tabulated and performance curves are given for these codes.

I. INTRODUCTION

It is known, if not widely disseminated, that the implementation of Viterbi decoders for high-rate convolutional codes is greatly simplified if the code structure is constrained to be that of a punctured low-rate code. In the standard approach to decoding these codes the implementation is complicated by the code structure which has 2^{n-1} paths entering each state rather than just two paths as rate $-1/n$ codes have. This makes the resulting comparison and selection of the path with the best metric much more difficult. The technique addressed here avoids this problem entirely. As a result one can decode just as one would decode a rate $-1/2$ code, with very little additional
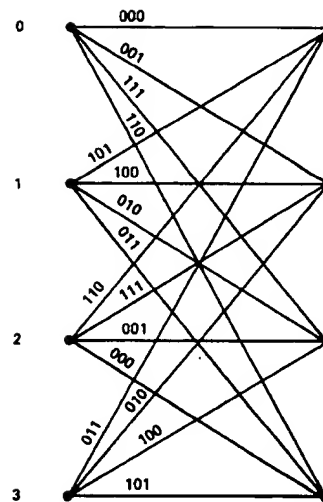
Fig. 1. Trellis structure for $R=2/3$, $\nu=2$ code.

complexity. A search for the best $R=2/3$ and $3/4$ codes of this type has been made. These codes are tabulated and performance curves are given.

II. PUNCTURED CODE STRUCTURE AND DECODING CONSIDERATIONS

Our discussion of punctured code structure uses the notation established by Forney [1]. The code constraint length is defined to be ν, the number of memory elements. A code is represented by its generator polynomial matrix $G(D)$. The element in the jth row and ith column,

$$G_j^i(D) = g_{0j}^i + g_{1j}^i D + \cdots + g_{\nu j}^i D^\nu, \tag{1}$$

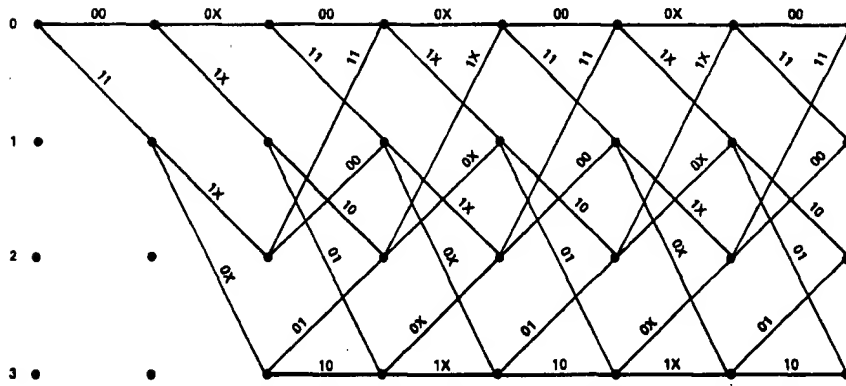relates the ith output sequence to the jth input sequence.

The punctured code approach will be illustrated using the $R=2/3$, $\nu=2$ code with generator matrix

$$G(D) = \begin{bmatrix} 1+D & 1+D & 1 \\ D & 0 & 1+D \end{bmatrix}. \tag{2}$$

The trellis structure for this code is shown in Fig. 1. In decoding this code using the Viterbi algorithm in the conventional manner, a 4-ary comparison must be made at each state, and one such 4-ary comparison per state must be made for every two information bits. This is in contrast to the much simpler binary comparisons performed in decoding $R=1/n$ codes.

Now consider the $R=1/2$, $\nu=2$ code with generators $1+D+D^2$ and $1+D^2$. If every fourth encoder output bit is deleted, this code will produce three channel bits for every two data bits; i.e., it will be a $R=2/3$ code. In fact, if the bit from the second generator $(1+D^2)$ is deleted from every other branch, the resulting code is identical to the $R=2/3$ code in our previous example. This code has the trellis shown in Fig. 2 where X indicates the deleted bits. Note that the transitions between states and the resulting transmitted bits are identical in Figs. 1 and 2, but in Fig. 2 the transition is through a set of intermediate states since only one bit at a time is shifted into the encoder rather than two. Obviously we have succeeded in generating the same code in a different manner.

In this example an $R=2/3$ code was constructed by periodically deleting bits from an $R=1/2$ code, or in other words by puncturing the code. Of course puncturing the code reduces its

Fig. 2.   Trellis diagram of $R=2/3$, $\nu=2$ code produced by periodically deleting bits from $R=1/2$, $\nu=2$ code.

TABLE I
GENERATORS (IN OCTAL) FOR THE BEST $R=2/3$ PUNCTURED
CODES WITH ONLY TWO DIFFERENT GENERATORS

ν	Generators	d	$w_i(d)$
2	7,5,7	3	1
3	15,13,15	4	8
4	31,33,31	5	25
5	73,41,73	6	75
6	163,135,163	6	1
7	337,251,337	8	395
8	661,473,661	8	97

TABLE II
GENERATORS (IN OCTAL) FOR THE BEST $R=3/4$ PUNCTURED
CODES WITH ONLY TWO DIFFERENT GENERATORS

ν	Generators	d	$w_i(d)$
2	5,7,5,7	3	15
3	15,17,15,17	4	124
4	25,37,37,37	4	22
5	61,53,53,53	5	78
6	135,163,163,163	6	919
6	121,165,121,165	5	21
7	205,307,307,307	6	117
8	515,737,737,737	6	12

TABLE III
GENERATORS (IN OCTAL) FOR THE BEST $R=2/3$ PUNCTURED
CODES

ν	Generators	d	$w_i(d)$
2	7,5,7	3	1
3	11,17,15	4	5
4	31,27,25	5	15
5	61,53,57	6	56
6	163,135,163	6	1
7	337,251,337	8	395
8*	441,767,565	8	47

*Partial search

TABLE IV
GENERATORS (IN OCTAL) FOR THE BEST $R=3/4$ PUNCTURED
CODES

ν	Generators	d	$w_i(d)$
2	5,7,5,7	3	15
3	15,17,15,17	4	124
4	31,27,23,23	4	10
5	61,47,65,65	5	51
6	165,127,117,173	6	276
7*	251,337,235,237	6	67
8*	631,557,431,455	6	10

*Partial search

TABLE V
COMPARISON OF FREE DISTANCES OF PUNCTURED CODES d_{fp} WITH
THOSE OF STANDARD CODES d_f

ν	R=2/3			R=3/4		
	Bound	d_f	d_{fp}	Bound	d_f	d_{fp}
2	4	3	3			
3	4	4	4	4	4	4
4	6	5	5	4	4	4
5	6	6	6	6	5	5
6	8	7	6	6	6	6
7	8	8	8	8	6	6*
8	8	8	8*	8	7	6*

*Partial Search

TABLE VI
SETS OF TWO DIFFERENT GENERATORS WHICH PRODUCE GOOD
$R=1/2, 2/3,$ AND $3/4$ CODES

ν	R=1/2 Generators(d)	R=2/3 Generators(d)	R=3/4 Generators(d)
2	7,5(5)	7,5,7(3)	7,5,5,7(3)
3	15,17(6)	15,17,15(4)	15,17,15,17(4)
4	31,33(7)	37,33,31(5)	31,33,31,31(3)
4	37,25(6)	37,25,37(4)	37,25,37,37(4)
5	57,65(8)	57,65,57(6)	65,57,57,65(4)
6	133,171(10)	133,171,133(6)	133,171,133,171(5)
6	135,147(10)	135,147,147(6)	135,147,147,147(6)
7	237,345(10)	237,345,237(7)	237,345,237,345(6)

free distance, in this case from 5 to 3. However, this distance is as large as can be achieved with any $R=2/3$, $\nu=2$ code. Paaske [2] made an exhaustive search of all generators and found no distance 4 codes. Thus in this case no loss in minimum distance is caused by using a punctured code.

This punctured code could also be thought of as an $R=1/3$ code with generators $1+D+D^2$, $1+D^2$, and $1+D+D^2$ that has the first two bits deleted on one branch and then the third bit deleted on the next branch. In fact it is simpler to represent these codes by using the generator matrix of the basic $R=1/n$ code, i.e.,

$$G^*(D) = \{ G_1^*(D) \quad G_2^*(D) \quad \cdots \quad G_n^*(D) \}, \qquad (3)$$

and the best codes that we found will be represented in this way in the tables in the next section. The convention that we have adopted is that the outputs corresponding to $G_1^*(D)$ and $G_2^*(D)$ are transmitted on the same branch and the outputs corresponding to each of the other generators are successively transmitted on separate branches.

The code generators that will be given in this form can also be easily transformed into the standard form of the generator matrix $G(D)$. In fact, it is easily shown that $G(D)$ for a $R=2/3$ punctured code is

$$G(D)$$
$$= \begin{bmatrix} G_{1e}^*(\sqrt{D}) & G_{2e}^*(\sqrt{D}) & G_{3e}^*(\sqrt{D})/\sqrt{D} \\ \sqrt{D}\, G_{1o}^*(\sqrt{D}) & \sqrt{D}\, G_{2o}^*(\sqrt{D}) & G_{3o}^*(\sqrt{D}) \end{bmatrix},$$
$$(4)$$

where $G_{ie}^*(D)$ and $G_{io}^*(D)$ are the even and odd power parts, respectively, of $G_i^*(D)$. The restrictions imposed on the generators by puncturing the code can be seen from (4). These are that $g_{02}^1 = g_{02}^2 = 0$ (which insures that the first 2 bits on each branch depend only on the previous state and the first information bit on that branch) and $g_{r1}^3 = 0$. The latter condition insures that there will be two pairs of branches entering each state for which the third bits on the branches are identical. This, of course, allows half of these paths to be eliminated in the comparison at the intermediate node. Similar restrictions can be derived for a general $R=(n-1)/n$ code.

The practical value of the punctured code approach is obvious. One can implement an $R=2/3$ decoder as an $R=1/2$ decoder with additional control to stuff erasures in the locations of the deleted bits. After the erasures are stuffed, decoding proceeds just as if the code were an $R=1/2$ code. In this fashion we replace the complex 2^{n-1}-ary comparisons at each state by binary comparisons. In a high-speed decoder this simplification has a major impact on decoder complexity. Even at low data rates when a serial implementation is used, the standard approach requires a factor of $(2^{n-1}-1)/(n-1)$ more binary comparisons per decoded bit than the punctured code approach.

III. SEARCH FOR OPTIMUM CODES

An extensive search was performed to find the best punctured code generators at each constraint length ν. Our definition of "best" is that code with the best performance on the AWGN channel at a large signal-to-noise ratio. This asymptotic performance for an $R=(n-1)/n$ code is given by [4]

$$P_e \approx \frac{w_i(d)}{n-1} Q\left(\sqrt{2 \frac{d(n-1)}{n} \frac{E_b}{N_0}} \right), \qquad (5)$$

where d is the free distance of the code and $w_i(d)$ is the total input weight of all information sequences which produce weight d paths. All codes with the largest free distance were exhaustively searched to find the minimum $w_i(d)$. While this criterion does not guarantee that the code selected will be the best at any specified error rate, say 10^{-5}, the resulting codes have nearly optimal performance at interesting values of output bit error rate. Of course these codes have also been checked to verify that they are noncatastrophic.

Since $w_i(d)$ was used as a criterion of optimality in addition to d, it was not possible to obtain a set of rules for drastically limiting the search as was done by Paaske [2]. Only the more obvious techniques were used. These include elimination of duplicate codes and making the branches entering and leaving each state in the $R=1/n$ code be antipodal.

The search for the best codes was done in two phases. In the first phase the search was over all generators where there are only two different generators among $G_1^*(D), G_2^*(D), \cdots, G_n^*(D)$. This formulation has implementation advantages particularly at high speeds in that the codes may be implemented as a single $R=1/2$ code with periodically sampled generators. The generators found in this phase with the smallest values of $w_i(d)$ for $R=2/3$ and $R=3/4$ were selected and are displayed in Tables I and II.

In one case ($R=3/4$, $\nu=6$) the only distance 6 code found had a very large multiplier for the distance 6 paths. The best distance 5 code was found to be about 0.1 dB better at $P_e = 10^{-5}$ so this code is also shown.

In the second phase of the search all n generators were allowed to be different. However, in several cases the best codes still only had two different generators. An interesting example of this is the familiar $R=1/2$, $\nu=2$ code which was discussed earlier. The generators for this code produce not only the best $R=1/2$ code but also the best $R=2/3$ and $R=3/4$ punctured codes for $\nu=2$. Conceptually one could think of these codes either as punctured $R=1/n$ codes (as discussed previously) or as punctured $R=1/2$ codes with periodically varying generators. The second interpretation could be easily implemented in a low-speed serial machine where the assignments of code branches to state transitions could be stored in a read-only memory. The code generators found in this phase of the search are shown in Tables III and IV.

The free distances found for the punctured codes compare very favorably with those found by Paaske [2] in his search over all generators for the standard codes. A comparison is shown in Table V.

Paaske's search is exhaustive in the sense that in those cases in which no code was found whose free distance meets the bound, no such code exists. Note that in only two cases ($R=2/3$, $\nu=6$ and $R=3/4$, $\nu=8$) does the best punctured code found have free distance less than that which is achievable by the standard approach. (In one of these cases, only a partial search has been performed). In all other cases, the same free distance is achieved with punctured codes that is achieved with the standard codes. The Plotkin upper bound [5] on free distance at each value of ν is also shown. In addition, one can also rederive this bound assuming the punctured code structure, and it is almost always equal to the bound for the standard codes.

This concept can also be used to provide a high-speed selectable rate decoder. To do this efficiently one must find two unique generator polynomials which can simultaneously generate good $R=1/2, 2/3$, and $3/4$ codes. Thus the code rate can be changed simply by changing the manner in which the generators are sampled. Typically, there does not exist a pair of generators which can simultaneously provide the best punctured $R=1/2$, $2/3$, and $3/4$ codes (this only occurs for the $\nu=2$ generators). However, a list of "good" generators and the free distance d for each generator was compiled and is shown in Table VI.
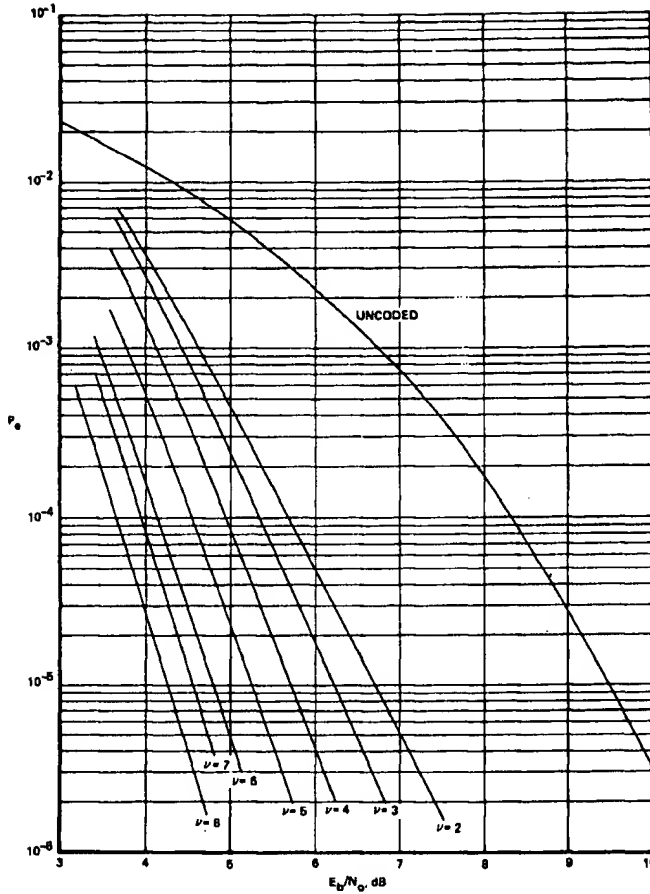
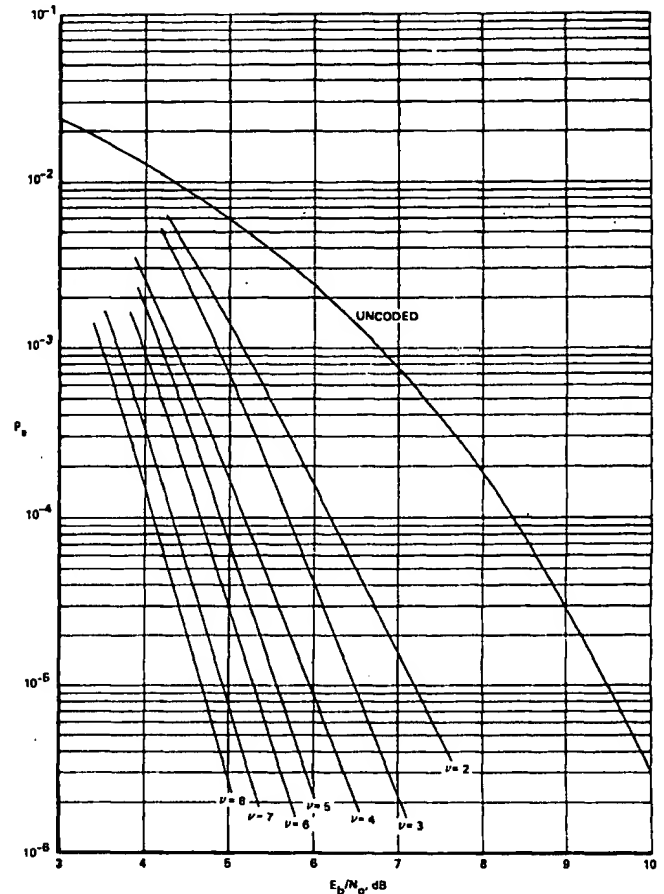Fig. 3. Performance of $R=2/3$ punctured codes with only two different generators.

Fig. 4. Performance of $R=3/4$ punctured codes with only two different generators.

IV. PERFORMANCE OF PUNCTURED CODES

Performance curves for the punctured codes were calculated using the union bounding technique. Using this method the bit error rate for an $R = m/n$ code is bounded by [3], [4]

$$P_e < \frac{1}{m} \sum_{j=0}^{\infty} w_i(j) P_j, \qquad (6)$$

where P_j is the probability that the correct path is eliminated by a merging path at Hamming distance of j, and $w_i(j)$ is the total information weight of all weight-j paths. We computed P_j for PSK signaling and AWGN using a method which accounts for the eight-level quantization customarily used. This technique produces union bound results which are found to agree with simulation results to within 0.1 dB at $P_e = 10^{-5}$.

The weight structure for the punctured codes was calculated including the terms up to weight 5 larger than the free distance of each code. These terms were then used in (6) to calculate performance curves. While only the curves for the codes with two different generators are shown in Figs. 3 and 4 (for brevity), we found that these codes are only slightly worse (0.1 dB or less) than the codes from Tables III and IV. This is fortunate for high-speed applications since the codes with only two different generators have decided advantages for implementation. Since Paaske's codes represent the most widely known list of standard codes, we also calculated union bounds for them and found that the longer codes were generally only about 0.1 dB better than the corresponding punctured codes. For the shorter constraint lengths, the Paaske codes are generally several tenths of a decibel better, due to the decreasing slope of the bit error rate

curve as ν decreases. Since Paaske's search was based on finding codes with optimal d_{free} rather than minimizing P_e, we recognize that there may exist standard codes with P_e somewhat better than that of Paaske's codes. However, in the range of interesting error rate, the potential improvement is very slight.

V. CONCLUSIONS

The purpose of this paper was to discuss a situation in which a difficult Viterbi decoder design problem can be simplified by suitably changing the code structure. The resulting decoder implementation is significantly simplified in high-speed applications. Fortunately, the restrictions on code structure implied by this approach result in only a slight performance loss (0.1 to 0.2 dB) in comparison with the best known $R = (n-1)/n$ codes. It is apparent from the implementation advantages and the performance curves that the $R=2/3$ and $R=3/4$ codes tabulated here have considerable practical significance.

REFERENCES

[1] G. D. Forney, Jr., "Convolutional codes I: Algebraic structure," *IEEE Trans. Inform. Theory*, vol. IT-16, pp. 720–738, Nov. 1970.
[2] E. Paaske, "Short binary convolutional codes with maximal free distance for rates 2/3 and 3/4," *IEEE Trans. Inform. Theory*, vol. IT-20, pp. 683–688, Sept. 1974.
[3] G. C. Clark, R. C. Davis, J. C. Herndon, and D. D. McRae, "Interim report on convolution coding research," Advanced Systems Operations, Radiation Systems Division, Memorandum Report No. 38, Sept. 1969.
[4] A. J. Viterbi, "Convolutional codes and their performance in communication systems," *IEEE Trans. Commun. Technol.*, vol. COM-19, pp. 751–772, Oct. 1971.
[5] J. A. Heller, "Sequential decoding: Short constraint length convolutional codes," *JPL Space Program Summary*, 37–54, vol. 3, pp. 171–174, Dec. 1968.